Part 3: Spatial Filtering

3.1 Implementation

- Applies a spatial, median, Gaussian and Edge Detection filter to an image
- Allow for input k to define the size of the filter
- Images remain in color and the filter is applied to the image of size m x n
- The resulting filtered images are saved as JPG files

3.2.1 Smoothing and Denoising

- A photo was applied with 3x3 and 27x27 sized Median and Gaussian filters for analysis

Figure 1: Original

Figure 2: Median 3x3 Filter

Figure 3: Gaussian 3x3 Filter

Figure 4: Median 27x27 Filter

Figure 5: Gaussian 27x27 Filter

When noise and resolution are compared, the pictures above show that the median filter is more effective in reducing noise in the photo while the Gaussian filter is more effective in maintaining the image's resolution. The median filtered photo produced a muddier looking image blending together the colors of the couch, clothes and dog. While there is not a noticeable difference in the photos after the 3x3 filter was applied, the 27x27 filter shows the difference in output. It is interesting however that both images introduce noise when a 3x3 filter is applied.

3.2.2 Edge Detection

Figure 1: Original image, no noise

Figure 2: Original image, with noise

Figure 3: Original image with edge detection

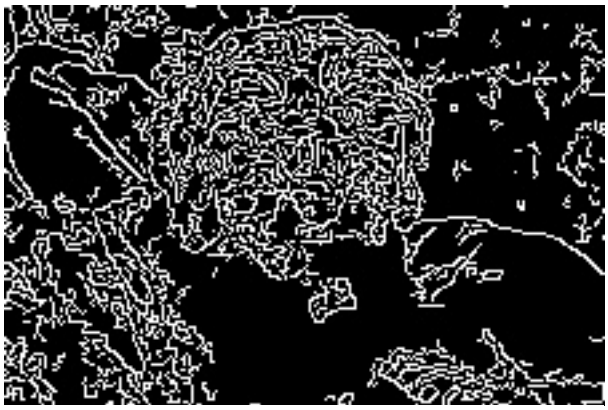


Figure 4: Original noisy image with edge detection

When comparing the effectiveness of edge detection on an image, it is evident that introduced noise leads to the Canny edge detection algorithm performing poorly. In the original image without introduced noise, the result has clearly defined edges around the dog's head and couch. However, the noise in the second original image introduces more "edges" in the result because the contrast between edges is not as apparent. The Canny algorithm is confused with the location of the edges because an even coat of noise is applied to the entire image creating drastic changes in pixel values near noisy pixels.

A solution to this problem would include smoothing a noisy image before applying an edge detection filter. That way, the difference of pixel values would not be so drastic and the algorithm would be better able to determine the "real" edges in the image.

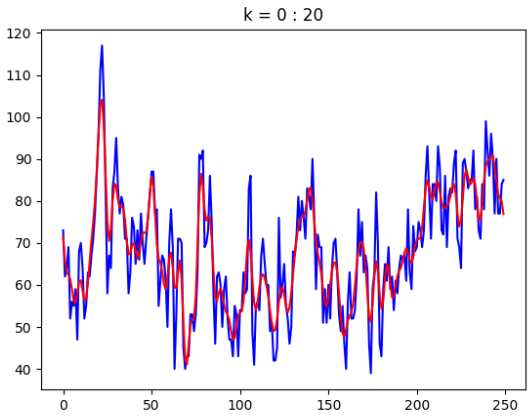
Part 4: Frequency Analysis

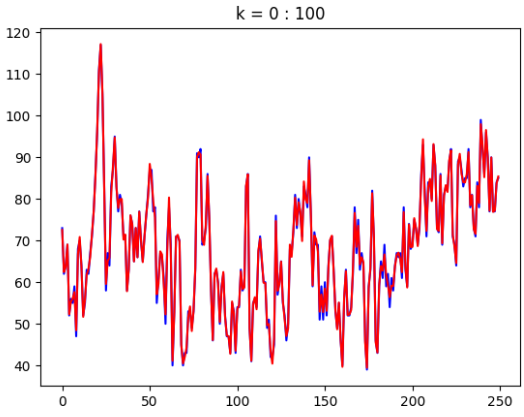
4.1 Implementation

- Creates a grayscale image of size m x n
- Visualizes the magnitude of the Fourier coefficients from the 2-D DFT image as 3-D plots
- Plots log+1 of magnitude of image
- Saves plots to JPG files

4.2 Frequency Analysis

Figure 1: Original image

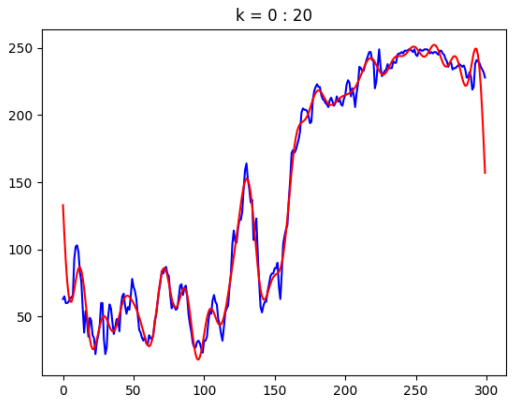


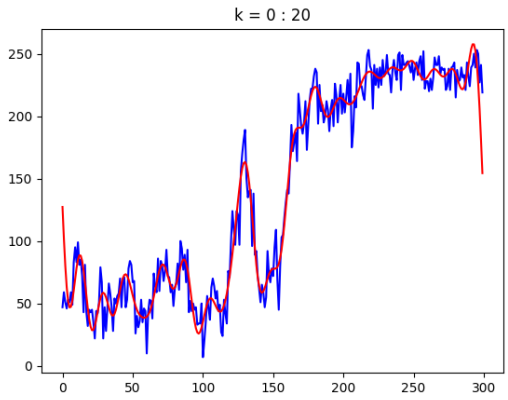


As seen above, the increasing Fourier coefficients correspond to a closer connection between the frequency data and the image column data. As the k value increases, the Fourier function becomes more accurate. From F(0) to F(100), the function slowly takes shape. This image is visually complex, so the Fourier function is more complex. A simpler image would have a simpler Fourier plot.

Figure 1: Original image

Figure 2: Original image, with noise





When comparing the Fourier functions of clear images vs. noisy images, we can see that the function is significantly more complex in the noisy image. The consistency of the curve is much less than the curve that correlates to the original image. For example, from 150 to 300 the graph corresponding to the noisy image on the right contains much more fluctuation and the k value would have to increase in order for the coefficient to better match the graph.

Often, a logarithmic form of data is used to reduce its skewness. The Fourier values (as seen plotted from above) will show one dominant value in the center of the plot. So, to better show the Fourier values in a top-down image we can take the logarithm + 1 of the values to create more balance in the resulting image. The image still contains a peak in the center but values in other part of the plot can also be seen. The magnitude plot and the log of the magnitude plot are below:

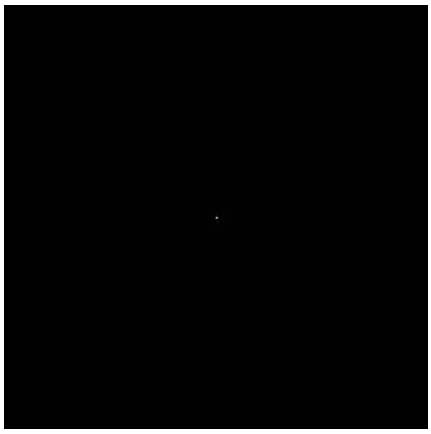


Figure 1: Magnitude plot

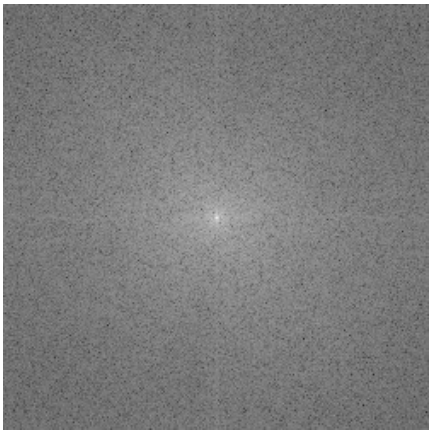
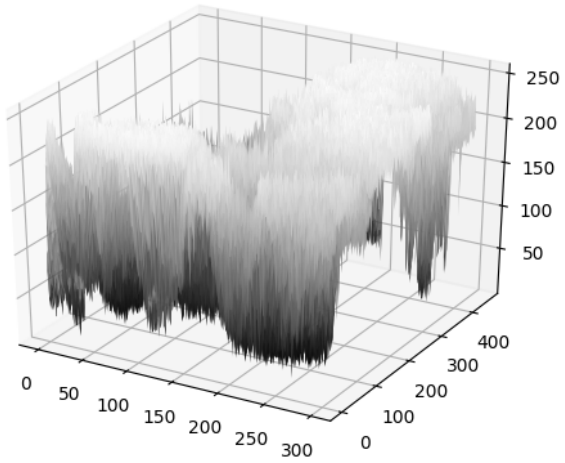
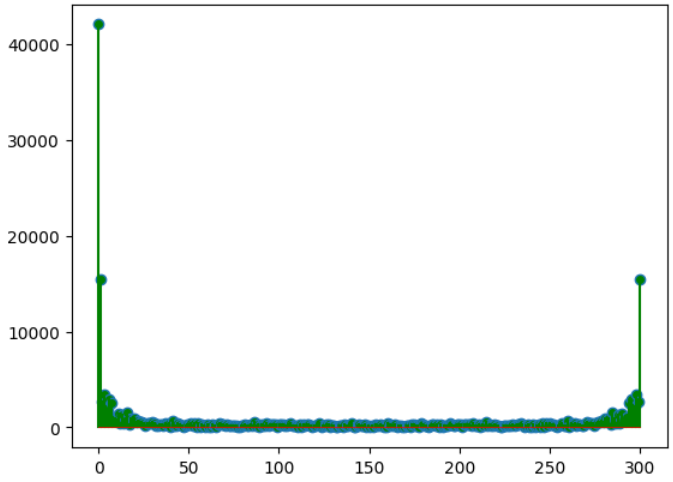


Figure 2: Log of Magnitude + 1





Seen above are the 3-D and 1-D visualizations of the Fourier transform function on the frequencies of the image. The 1-D image on the right represents the 2-D plots above. At 0, this different function has an extremely large peak. At 300 we see another peak although much smaller in magnitude. The 3-D image gives us a much more in-depth understanding of the frequencies across the entire image.

Part 5: Frequency Filtering

5.1 Implementation

- Creates grayscale image of size m x n
- Computes the 2-D DFT of the image
- Dampens Fourier coefficients with Butterworth's approach
- Plot original and Butterworth-scaled magnitudes of Fourier coefficients and their images
- Save plots in JPG files

5.2 Low-pass and High-pass Filtering

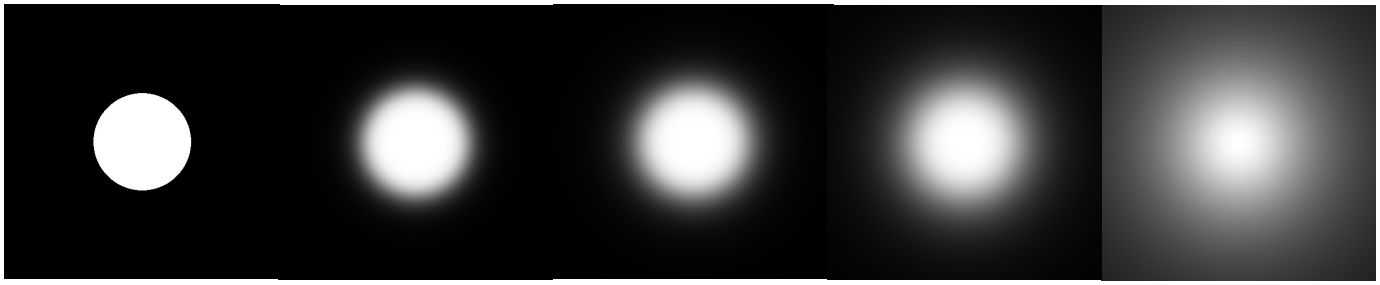


Figure 1: Ideal Low Pass, N(1), N(2), N(3), N(4)

Above is an ideal low-pass filter without an image. This filter dampens the Fourier coefficients in order to produce a different image result. We will use this figure and compare it to an image with the filter applied.

N(1)

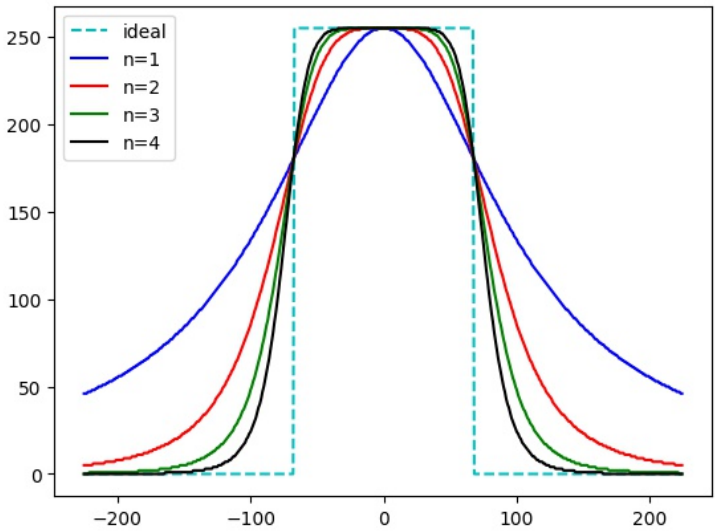
N(2)

N(3)

N(4)

Ideal Low-Pass Filter

In these photos there isn't a significant difference between the filtered images, however when comparing N(1) to N(4) we can see that the clarity and detail in N(1) is more apparent than in N(4). But the photo with the least amount of clarity and detail is the ideal low-pass filtered image.



Looking at the graphical representation of the low pass filters, as the n values increase they arrive closer to the ideal dotted line. N = 4 is the closest line to the ideal low pass Butterworth filter.

It is important to understand why low-pass filters are used. They allow low frequencies to remain while lowering the magnitude of the high frequencies within an image. In an image with high frequency, this filter is effective in reducing noise.